Incremental processing and the hierarchical lexicon

van der Linden, H.J.B.M.

Publication date:
1992

Document Version
Publisher's PDF, also known as Version of record

[redacted]

ITK

RESEARCH REPORT

INSTITUTE FOR LANGUAGE TECHNOLOGY AND ARTIFICIAL INTELLIGENCE

INSTITUUT VOOR TAAL- EN KENNISTECHNOLOGIE

ITK Research Report No. 36

Incremental Processing and the Hierarchical Lexicon

Erik-Jan van der Linden

ITK
Warandelaan 2
P.O. Box 90152
5000 LE TILBURG
vdlinden@kub.nl

May 1992

Incremental Processing and the Hierarchical Lexicon

Erik-Jan van der Linden[1]

Tilburg University

Abstract

Hierarchical lexicon structures are not only of great importance for the nonredundant representation of lexical information, they may also contribute to the efficiency of the actual processing of natural language. Two parsing techniques that render the parsing process efficient are presented. *Windowing* is a technique for incrementally accessing the hierarchical lexicon. *Lexical Preferencing* implements preferences within the parsing process as a natural consequence of the hierarchical structure of the lexicon. Within a proof-theoretic approach to Categorial Grammar it is possible to implement these techniques in a formal and principled way. Special attention is paid to idiomatic expressions.

1 Introduction

The main reasons mentioned for the considerable attention paid to hierarchical lexicon structures are the fact that redundancy in the lexicon is avoided, and that structuring the lexicon facilitates the development of large and complex lexicons. No attention has, however, been paid to the role the hierarchical lexicon could play in natural language processing. Categorial Grammar (CG) has an interest in efficient and psychologically plausible, at least incremental, processing. Although CG is a radically lexicalist grammatical theory, little attention has been paid to the structure of the lexicon. The aim of the present article is to bring CG, the hierarchical lexicon and incremental processing together, to investigate the role of the hierarchical lexicon during incremental parsing with categorial grammars. The rules and derivations of a categorial grammar do not describe syntactic structures, but represent the proceedings of the parser while constructing a semantic representation of a sentence. This property of CG is referred to as *representational nonautonomy* (Crain and Steedman 1982). It will be shown that especially in the case of ambiguity, the combination of a hierarchical lexicon structure and representational nonautonomy provides efficient ways of dealing with ambiguities: within a proof-theoretic approach to CG, rules are presented which allow the parser to reason about the structure of the lexicon. Two parsing techniques are presented. *Windowing* is a technique for incrementally accessing the hierarchical lexicon. While incrementally parsing the sentence, the parser commits itself to lexical information it can commit to, leaving other choices implicit in the hierarchical lexical structure of the elements in the input. *Lexical Preferencing* implements preferences in the parsing process as a natural consequence of the hierarchical structure of the lexicon: information lower on in the hierarchical lexicon is preferred over more general information. Idiomatic expressions are presented as an example of these preferences: an idiomatic expression is preferably interpreted as such, and not in the non-idiomatic interpretation of which the head of the idiom is a part.

In Section 2 the proof-theoretic approach to CG is presented. Next, in Section 3 a hierarchical lexicon structure for CG is presented. Two category-forming connectives that make the structure of the lexicon visible for the parser are introduced. Windowing is discussed in Section 4. In Section 5 parsing preferences are discussed in general, and preferences for the interpretation of idioms in particular.

[1]Institute for Language Technology and AI (ITK), PO Box 90153, 5000 LE Tilburg, The Netherlands. E-mail: vdlinden@kub.nl

2 Categorial Grammar and Proof Theory

2.1 The Lambek calculus

Recently, proof theory has aroused the interest of categorial grammarians. In the Lambek calculus (**L**-calculus, **L**; Lambek 1958), the most salient example of the application of proof theory to categorial grammar, the rules of the grammar become a set of axioms and inference rules. Together, these form a logical calculus in which parsing of a syntagm is an attempt to prove that it follows as a theorem from the set of axioms and inference rules. Following the work of Van Benthem (1986) and Moortgat (1988, 1987) the Lambek calculus has become popular among a number of linguists (Barry and Morrill 1990; Hendriks 1987).

Categories in CG can either be basic (*np*, *s*, *n*), or complex. A complex category consists of a binary category-forming connective and two categories, for instance $np\backslash s$. In the product-free **L**-calculus the set of connectives (also called type constructors), is $\{\backslash, /\}$. A complex category is a functor, an incomplete expression which forms a result category if an argument category is found. Throughout this paper the Lambek notation in which the argument category is found under the slash is applied. Consider for example the categorial representation of an intransitive verb: $np\backslash s$ looks for an *np* to its left and results in an *s*.

The elements the calculus operates upon are categories with semantic and prosodic information added, denoted with $< syntax, prosody, semantics >$, and referred to as signs. Information not relevant for the discussion is omitted. In the version of **L** used here, complex syntactic categories take signs as their arguments. Semantics is represented with formulas of the lambda-calculus. Prosodic information merely consists of a prosodic bracketing; for instance the string *john sleeps* is denoted as $john + sleeps$, where $+$ is a noncommutative, nonassociative concatenation operator. Concatenation of some ϕ with the empty prosodic element ε results in ϕ.

The **L**-calculus extends the power of categorial grammar basically because it adds so-called *introduction* rules to the proof-theoretic complements of categorial reduction rules, *elimination* rules. For each category forming connective, introduction and elimination rules can be formulated. With respect to semantics, elimination corresponds to functional application and introduction to lambda abstraction. Various approaches have been proposed for deduction in **L**. In its standard representation the **L**-calculus is a *sequent* calculus. More recently, natural deduction has been applied to the calculus (Barry and Morrill 1990), as well as proof procedures from linear logic (Roorda 1990).[2] Throughout this article the sequent format is used.

In definition (1), W and X are categories, Y and Z are signs, and P, T, Q, U and V are sequences of signs, where P, T and Q are nonempty. A sequent in **L** represents a derivability relation, $\Rightarrow$, between a nonempty finite sequence of signs, the antecedent, and a sign, the succedent. A sequent states that the string denoted by the antecedent is in the set of strings denoted by the succedent. The axioms and inference rules of the calculus define the theorems of the calculus with respect to this derivability relation. Recursive application of the inference rules on a sequent may result in the derivation of a sequent as a theorem of the calculus. In definition (1) the calculus is presented. The elimination of a type constructor is denoted by E, introduction by I.

[2] For a comparison, see Leslie (1990).

Definition 1 (Lambek, sequent calculus)

U,<(X/<W,ψ,b>),ϕ,a>,T,V ⇒ Z [/E]
if T ⇒ <W,ψ,b>
and U,<X,ϕ+ψ,a(b)>,V ⇒ Z

U,T,<(<W,ψ,b>\X),ϕ,a>,V ⇒ Z [\E]
if T ⇒ <W,ψ,b>
and U,<X,ψ+ϕ,a(b)>,V ⇒ Z

T ⇒ <(X/<W,e,b>),ψ,λb.a> [/I]
if T,<W,e,b> ⇒ <X,ϕ+e,a>

T ⇒ <(<W,e,b>\X),ϕ,λb.a> [\I]
if <W,e,b>, T ⇒ <X,e+ϕ,a>

<X,ϕ,a> ⇒ <X,ϕ,a> [Axiom]

The uppersequent of an inference rule is a theorem of the calculus if all of its subsequents are theorems. In example (1) a sentence containing a transitive verb is parsed by proving that it reduces to *s*. To the sequence of lexical signs associated with the strings in the input, the inference rules are recursively applied until all leaves of the proof tree are axioms. The derivation results in the instantiation of the semantics of the sentence.

(1)

<np,john,john> <(np\ s)/np,loves,loves> <np,mary,mary> ⇒ <s,john+loves+mary,loves(mary)(john)> [/E]
if <np,mary,mary> ⇒ <np,mary,mary> [Axiom]
and <np,john,john> <np\s,loves+mary, loves(mary)> ⇒ <s,john+loves+mary,loves(mary)(john)> [\ E]
if <np,john,john> ⇒ <np,john,john> [Axiom]
and <s,john+loves+mary,loves(mary)(john)> ⇒ <s,john+loves+mary,loves(mary)(john)> [Axiom]

2.2 Other connectives

The product-free version of the Lambek calculus includes two connectives, / and \. On the basis of these connectives and the inference rules of the **L**-calculus, a range of linguistic constructions and generalisations remain for which no linguistically adequate accounts can be presented. In order to overcome this problem, new category-forming connectives have been proposed (as a lexical alternative of the specialized rules of for instance CCG (Steedman 1987)). An example is the connective ↑ for unbounded dependencies (Moortgat 1988). $X \uparrow Y$ denotes a category X that has an argument of category Y missing somewhere within X. The constituent *John put on the table* in *what John put on the table* has as its syntactic category $s \uparrow np$. To *what* the category $s/(s \uparrow np)$ is assigned, which takes the incomplete clause as its argument.
The $\wedge$-connective (Morrill 1990) is one of a set of boolean connectives that can be used to denote that a certain lexical item can occur in different categories: *square* can be n/n and n, and is therefore assigned the category $(n/n) \wedge n$. The ?-connective (ibid.) is used to denote optionality, for instance in the case of *belief*: $n/(sp?)$ which accounts for *belief* in *the belief* and *the belief that Mary lives*.

These connectives are introduced to enable the inference engine behind the calculus to deal with lexical ambiguities and to 'reason' about lexical items. This is in line with the principle of representational nonautonomy which states that syntactic rules describe *what* the processor does while assembling a semantic representation.

3 Inheritance and the hierarchical lexicon

To allow the inference engine to reason about lexical structures in which inheritance relations are present, the calculus should be extended, and a more sophisticated structure should be assigned to

the categorial lexicon than the list or bag it is usually considered in CG (with the exception of Bouma (1990)). The current section deals with the lexicon, the next sections deal with the extension of the calculus.

3.1 An example: idioms

An idiomatic expression and its verbal head can be said to maintain a lexical inheritance relation: an idiomatic expressions inherits part of its properties from its head. Here, syntactic category, syntactic behaviour, morphology and semantics are discussed briefly.

Syntactic category Idiomatic expressions can be represented as functor-argument-structures[3] and have the same format as the verbs that are their heads. It is therefore possible to relate the syntactic category of the idiom to that of its head (see also Zernik and Dyer (1987)). The verb itself does not specify prosodic information for the argument and the idiom is a specialisation of the verb because it does specify prosodic information. In other words, the verb (*kick*) subcategorizes for the whole set of strings with category *np*, whereas the idiom subcategorizes for the subset of that set (*the+bucket*). The information that the object argument is specified for a certain string can thus be added monotonically. Inheritance relations between lexical items are denoted here with a category-forming connective $\succ$. $Mother \succ Daughter$ states that *Daughter* is a specialisation of *Mother*. The relation between verb and idiom is part of the lexical structure which is associated with the lexical entry of the verb. KICK, KICK_TV and KICK_THE_BUCKET are represented as in (2).

(2) a. KICK: $< KICK\_TV \succ KICK\_THE\_BUCKET, kick >$
b. KICK_TV: $< (np\backslash s)/ < np, \_ >, \_ >$
c. KICK_THE_BUCKET: $< \_/ < \_, the + bucket >, \_ >$

Morphological properties The verb that is the head of an idiomatic expression has the same inflectional paradigm as the verb outside the expression: for instance, if a verb is strong outside an idiom, it is strong within the idiom.

Syntactic behaviour The syntactic behaviour of idioms should partly be explained in terms of properties of their heads. For example, it is not possible to form a passive on the basis of predicative and copulative verbs, either inside or outside an idiomatic expression.[4] This information is inherited by the idiom from its verbal head.

Semantics The traditional definition of an idiom states that its meaning is not a function of the meanings of its parts and the way these are syntactically combined, that is, an idiom is a noncompositional expression. Under this definition, their meaning can be subject to any other principle that describes in what way the meaning of an expression should be derived (contextuality, meaning postulates...). A definition that states what the meaning *is*, is preferable: the meaning of an idiom is exclusively a property of the whole expression.[5] The meaning of the idiom cannot be inherited from the verb that is its head, but should be added nonmonotonically.

(3) a. KICK: $< KICK\_TV \succ KICK\_THE\_BUCKET, kick, \lambda x \lambda y kick(x)(y) >$
b. KICK_TV: $< (np\backslash s)/np, \_, \_ >$
c. KICK_THE_BUCKET: $< \_/ < \_, the + bucket, \_ >, \_, \lambda x \lambda y die(y) >$

[3] Similar representations can be found for TAG (Abeillé 1990; Abeillé and Schabes 1989) and HPSG (Erbach 1991).
[4] See van der Linden (1991).
[5] See van der Linden and Kraaij (1990) and van der Linden (1991) for a more extensive comparison of this definition and the traditional one.

Inheritance The full specification of a sign is derived by means of an operation similar to ***priority union*** (Kaplan 1987:180) or ***default unification*** (Bouma 1990), denoted by $\cap$. $\cap$ is defined as a function from pairs of mother and daughter signs to fully specified daughter signs and runs as follows. If unification, $\cup$, is successful for the values of a certain property of mother and daughter, the result of $\cap$ for that value is the result of $\cup$, where unification is understood in its most basic sense: variables unify with constants and variables; constants unify with variables and with constants with an equal value (prosodic information in (4)). If the values do not unify, the value of the daughter is returned (semantic information in (4)).

(4) $(KICK \cap KICK\_TV) \cap KICK\_THE\_BUCKET$:
$< (np\backslash s)/ < np, the + bucket, \_ >, kick, \lambda x \lambda y die(y) >$

The inheritance networks for which $\cap$ is defined are unipolar, nonmonotonic and homogeneous (Touretzky et al. 1987). For other networks, other reasoning mechanisms are necessary to determine the properties of a node (Touretzky et al. 1987; Touretzky 1986; Veltman 1990).[6]

More specific information thus takes precedence over more general information. This is a common feature of inheritance systems, and is an application of 'proper inclusion precedence' which is acknowledged in knowledge representation and (computational) linguistics (De Smedt 1990; Daelemans 1987; other papers in these special issues).
There exists a clear relation between this principle and the linguistic notion *blocking*. Blocking is "the nonoccurrence of one form due to the simple existence of another" (Aronoff 1976:41). For instance the nominal derivation **graciosity* of *gracious* is blocked by the existence of *grace*. Daelemans (1987) and De Smedt (1990) show that in a hierarchical lexicon structure, blocking is equivalent to the prevalence of more specific information over more general information. For instance, the more general principle in the example is that a nominal derivation of some abstract adjectives equals *stem + ity*, and the more specific information is that in the case of *gracious* the nominal derivation is *grace*. In the hierarchical lexicon, proper inclusion precedence also blocks ?*graciousness* (whereas this is not the case for Aronoff's model). In Dutch, the participle **geslaapt* that has been formed on the basis of regular morphological processes is blocked because the past participle of *slapen* is *geslapen*.

3.2 Other lexical relations

Verbs that can be either transitive or intransitive, like *kick*, can in principle be modelled with the use of the $\wedge$-connective: $< np\backslash s, \lambda y \exists x kick(x)(y) > \wedge < (np\backslash s)/np, \lambda x \lambda y kick(x)(y) >$.
There are, however, two generalisations missing here. Firstly, the transitive and the intransitive form share the syntactic information that their reducible category is s and their subject argument is np. Secondly, the denotation of the transitive subsumes the denotation of the intransitive: the semantics of the transitive verb is more specific than the semantic representation of the intransitive. The use of the optionality operator ? ($((np\backslash s)/?np)$) would imply that *kick* is in principle an intransitive verb, that has one optional argument, whereas in fact the reverse is true: *kick* is a two-place-functor of which one argument may be left unspecified syntactically. The transitive and intransitive verb can be said to share their semantic value, but in the case of the intransitive, the syntactically unspecified object is not bound by a λ-operator but by an (informationally richer) existential quantor. The transition from the transitive to the intransitive is represented as a lexical type-transition (Dowty 1979:308).

Definition 2 (detransitivisation)
detrans:
$\lambda x \lambda y D(x)(y) \Rightarrow \lambda y \exists x D(x)(y)$

From a syntactic point of view, the transitive form of the verb can be said to inherit the syntactic information from the intransitive and to add a syntactic argument. From a semantic point of view,

[6] Touretzky (1986) also discusses default logic and nonmonotonic logic.

the transitive inherits the semantic information that is specified for the KICK entry as a whole. The intransitive inherits the same information, and stipulates application of detransitivisation to it. The lexical relation between the transitive and the intransitive is thus different from that between a verb and an idiom: in the case of the idiom a syntactic argument is further instantiated whereas here a syntactic argument is *added*. In order to represent this distinction, a different connective is used: $\gg$. With the use of this type constructor, the intransitive and the transitive can be placed in an inheritance relation (5). $\gg$ is a category forming connective which takes two signs to form a category.

(5) a. KICK: $< KICK\_IV \gg KICK\_TV, kick, \lambda x \lambda y kick(x)(y) >$
c. KICK_IV: $< np\backslash s, \_, detrans(KICK) >$
b. KICK_TV: $< synt(KICK\_IV)/np, \_, sem(KICK) >$

The lexical structure presented here can be considered equal to that presented by Flickinger (1987) and Pollard and Sag (1987) for HPSG. They present a hierarchy in which not only transitive and intransitive verbs, but other classes of verbs are represented as well. A minor difference is that Flickinger and Pollard and Sag place classes of verbs in hierarchical relations, whereas here individual verbs maintain inheritance relations. The main difference with this and other previous approaches is that with the introduction of connectives for inheritance relations, inference rules for these connectives can be presented that describe the legal moves of the inference engine when reasoning about these lexical structures. This will be discussed in the next section.

4 Windowing

4.1 Incrementality and immediacy

Left-to-right, incremental processing contributes to the speed of the parsing process because parts of the input are processed as soon as they are encountered, and not after the input has been completed. Besides, because of the fact that processing is incremental, it is possible to give an interpretation of a sentence at any moment during the parsing process.[7]
Immediate interpretation, which entails that the processor deals with semantics as nearly as possible in parallel with syntax, contributes to the efficiency of the interpretation process because ambiguities are solved as soon as possible and processing downstream is thus not bothered with alternative analyses. Categorial grammar enables incremental and immediate processing since categorial grammar allows for flexible constituent structures: any two signs can be combined to form a larger informational unit. For a parsing process to be incremental, it should reduce two constituents if these maintain a head-argument relation. The incremental construction of analyses for sentences with the use of phrase structure grammars is not in all cases possible. For example, in case the input consists of a subject and a transitive verb it is only possible to integrate these two into a sentence if the object has been parsed: only then can the *vp* be formed and combined with the subject to form an *s* (Briscoe 1987). A process like this cannot be called incremental. Although subject and verb can be processed incrementally independently from each other, this is not the case for their combination.
The strategy mostly used in incremental CG-processing is to enable the construction of a semantic structure with the use of principles that concatenate *all* possible adjacent categories (although some exceptions are made for coordinate structures (Dowty 1988; Houtman 1987)). In Combinatory Categorial Grammar (Ades and Steedman 1982; Steedman 1987), for instance, composition and lifting rules (definition 3 and 4) enable incremental interpretation (example 6). To make use of these rules in a proof-theoretic approach to CG, a rule is necessary which *cuts* the result of these rules in the proof as a whole (5).

[7]The first parser that featured incremental processing can be found in Marcus (1980). This parser did not consider lexical ambiguity, and confined itself to syntactic processing. Other computational models that entail the notion of incrementality can be found for Segment Grammar and in Word Expert Parsing. The subsymbolic processing architecture for Segment Grammar presented by Kempen and Vosse (1989) is a model of syntactic processing. The architecture allows for immediate interpretation, but no semantic representation is actually constructed. Adriaens (1986) presents a lexicalist model, Word Expert Parsing, which operates incrementally. Another lexicalist model that features incremental processing can be found in Stock (1989). In none of the models is mention made of a structured lexicon.

Definition 3

<X/Y,a> <Y/Z,b> ⇒ <X/Z, λx.a(b(x))> [Comp]

Definition 4

<X,a> ⇒ <Z/(X\Z), λb.b(a)> [Lift]

Definition 5

U, X, Y, V ⇒ Z [Cut]
if X, Y ⇒ W
and U, W, V ⇒ Z

(6)

<np,john> <(np\s)/np,kicks> <np/n,the> <n,boy> ⇒ <s, kicks(the(boy))(john)> [Cut]
if <np,john> ⇒ <s/(np\s), λX.X(john) > [Lift]
and <s/(np\s), λX.X(john) > <(np\s)/np,kicks> <np/n,the> <n,boy> ⇒ <s, kicks(the(boy))(john)> [Cut]
if <s/(np\s), λX.X(john) > <(np\ s)/np,kicks> ⇒ <s/np, λx.kicks(x)(john)> [Comp]
and <s/np, λx.kicks(x)(john)> <np/n,the> <n,boy> ⇒ <s, kicks(the(boy))(john)> [Cut]
if <s/np, λx.kicks(x)(john)> <np/n,the> ⇒ <s/n, λy.kick(the(y))(john)> [Comp]
and <s/n, λy.kick(the(y))(john)> <n,boy> ⇒ <s, kicks(the(boy))(john)> [/E]
if <n, boy> ⇒ <n,boy> [Axiom]
and <s, kicks(the(boy))(john)> ⇒ <s, kicks(the(boy))(john)> [Axiom]

All words, also the function words like *the* are in principle processed and thus interpreted immediately, that is, their semantic representation is accessed from the lexicon and combined with the semantic representation of the input so far.
A similar proposal is the **M**-calculus (Moortgat 1988, 1990). In **M**, the Elimination rules of **L** are traded in for a set of *generalized application* rules and a cut-rule which links the derivation relation ⇒ and the derivation relation of the system of generalized application, ⇒* (6).

Definition 6

<X/<Y,ψ,a>,φ,b>, <Z,χ,c> ⇒* <X,φ+ψ,b(a)> [M1/]
if <Z,χ,c> ⇒ <Y,ψ,a>

<Z,χ,c>, <<Y,ψ,a>\X,φ,b> ⇒* <X,ψ+φ,b(a)> [M1\]
if <Z,χ,c> ⇒ <Y,ψ,a>

<Z,ψ,c>, <X/Y,φ,b> ⇒* <W/Y,ψ+φ,λa.d> [M2/]
if <Z,ψ,c>, <X,-,b(a)> ⇒* <W,-,d>

<Y\X,φ,b>, <Z,ψ,c> ⇒* <Y\W,ψ+φ, λa.d> [M2\]
if <X,-,b(a)>, <Z,ψ,c> ⇒* <W,-,d>.

<X/Y,φ,b>, <Z,ψ,c> ⇒* <X/W,φ+ψ,λd.b(a)> [M3/]
if <Z,ψ,c>, <W,-,d> ⇒* <Y,-,a>

<Z,ψ,c>, <Y\X,φ,b> ⇒* <W\X,ψ+φ,λd.b(a)> [M3\]
if <W,-,d>, <Z,ψ,c> ⇒* <Y,-,a>

U,<X,φ,a>,<Y,ψ,b>,V ⇒ <Z> [M-Cut]
if <X,φ,a>,<Y,ψ,b> ⇒* <Cut,χ,c>
and U,<Cut,χ,c>,V ⇒ <Z>

M is also capable of processing a sentence in an incremental fashion, as each word is added to the semantic structure as it is encountered. These Categorial Grammars thus implement incrementality and an all-or-none immediacy: there is at all times during the parsing process a full interpretation of the input so far.[8]

[8] In the **P**-calculus (Bouma 1989) a shift-reduce strategy is modelled for a categorial parser. Reduction corresponds to the application of a categorial reduction rule; a shift is represented by connecting two categories by means of the product operator '*'. During later stages of incremental processing this product formula is taken apart, and the parts are used for constructing a semantic representation. Bouma notes with respect to his **P**-calculus that connecting two semantic representations with a '*' can hardly be called building up a semantic representation.

There are two problems with this approach. Firstly it is questionable whether it agrees with the psycholinguistic notion of immediacy, and secondly it leads to an unrealistic view of the parsing process. The second point will be discussed in the following section.
With respect to the first point of criticism it should in the first place be noted that immediacy as it was formulated by Just and Carpenter (1980) was only formulated for *content* words. The immediacy assumption states that processing of *content* words should be as immediate *as possible*. Firstly, CG has included *function* words under immediacy. Haddock (1987),[9] for instance, states that given a domain with two rabbits that are 'in' something, during incremental processing of the phrase *the rabbit in the hat*

> "the incremental evaluation of *the rabbit in the* has created two distinct sets of candidates for the two NPs in the phrase" (Haddock 1987:81)

It is not clear from the psycholinguistic literature whether processing of function words takes place this way, but it is at least unintuitive: in larger domains large intermediate sets of candidates will be of little help for the interpretation in comparison to the information the constituent as a whole provides. Secondly, the wish to be able to give an interpretation of a sentence at *any* stage of the parsing process stems from the fact that humans are able to make guesses about continuations of sentences that stop before they have come to a proper ending (Schubert 1984). From this it follows that humans *are able to* construct interpretations at any moment during NLP, but not that they actually *do* construct full interpretations: the ability to complete incomplete sentences says little about the ongoing automatic interpretation process.

4.2 Windowing and Lexical ambiguity

The $\gg$-operator is useful for incremental processing in case of lexical syntactic ambiguity and overcomes one of the problems of all-or-none immediacy.
One of the sources of lexical ambiguity is that a functor may have several subcategorisation frames. During incremental processing, one of the subcategorisation frames of an ambiguous word has to be selected. *How* this choice is made is unclear in most categorial work that claims to model incremental processing: ambiguity is not an issue.[10] With the use of operators like $\wedge$ the ambiguity can at least be described, but truly incremental processing does not seem possible: the all-or-none immediacy leads to a unrealistic parsing process. An example will illustrate this. In (7) part of the derivation of *John gave a book to Mary* is presented.

(7)

John	gave	
np	$((np\backslash s)/pp)/np \wedge (np\backslash s)/pp \wedge (np\backslash s)$	
	.....	

The problem that faces the parser here is that it is forced to choose one of the subcategorisation frames in order to make this step in the derivation. There is, however, no indication which frame should be selected. In the case an incorrect frame is selected, backtracking is necessary when further material in the input is contradictory with this frame. For instance, the choice of a frame without a direct object will lead to a semantic representation which includes the binding of the object position by an existential quantifier. If the parser later on encounters a direct object, this will lead to a revision of the choice of the category of *give*, and to revision of the interpretation of the sentence.

After having encountered *John gave*, the parser only has to commit itself to the fact that there is (at least) one *np*-argument to the verb, that the result category is s, and that this argument semantically functions as the subject: $\lambda x \lambda y give(x)(y)(john)$. Whatever the continuation may be, intransitive,

[9] Haddock (1987) proposes a 'reduce-first' strategy for the incremental categorial parsing. It

> "(...) will always reduce, remembering the shift option as an alternative which could be chosen in the event of backtracking" (Haddock 1987:75)

[10] Ades and Steedman (1982) state this explicitly.

transitive or ditransitive, this semantic representation subsumes the semantics of the whole sentence. Whether the continuation is intransitive, transitive or ditransitive cannot be decided, and should be left unspecified. In terms of the hierarchical relation between the frames as it was linguistically motivated in the previous sections (compare the inheritance hierarchy for *kick* in (3)) the parser should commit itself to the information which is valid for the inheritance hierarchy as a whole, and to the syntactic information of the intransitive form, but it does not yet have to commit itself any further. The parser can, while incrementally processing a sentence, keep a *window* on the lexical structure, which becomes smaller iff there is evidence in the input that one of the frames is the right frame. Since parts of the information are shared among the different frames, information once gained is not lost, but is available for all frames. This technique of careful incremental lexicon access will be referred to as *windowing* here. It can be considered a syntactic counterpart of the semantic *Polaroid Words* of Hirst (1988) for which the meanings become more specific (develop) in the light of evidence in the input, except that Polaroid Words are active objects.

Since the hierarchical structure of the lexicon can be made visible to the parser by means of the $\gg$-operator, it is possible to model windowing by means of the inference rules for the $\gg$-operator. In definition (7) elimination rules for $\gg$ are presented. [11] Together with the **M**-system, these rules form a calculus which enables incremental processing and incremental access to the lexicon. It will be referred to as the **I**-calculus (**I** for inheritance), and will be used in what follows.[12]

To link derivability in the **L** calculus to $\cap$, $\cap >$ relates a *node* from a *hierarchy* to its *specification*: (*hierarchy*, *node*) $\cap >$ *specification*. *Node* can either be the top-node of the hierarchy, *mother*, its *daughter*, or the *granddaughter* which is the node in the hierarchy that is linked to its mother with the $\gg$-operator, but which has no $\gg$-daughters.

The inference rule that eliminates the inheritance operator has three instances. In first case, the sign on top of the lexical hierarchy combines with an argument sign in the input (this rules has a right-looking counterpart). In the second case, the result of the elimination of $\gg$ is the daughter. In the third case, the result is the mother. In line with representational nonautonomy these rules describe what the processor does while assembling a semantic representation. In example 8 an example is presented. The prosodic terms are left out for reasons of clarity.

Definition 7 (Inference rules for $\gg$)

T, <<(mother_arg\ mother_result),sem_mother>$\gg$ <syn_daughter,sem_daughter)>, sem>, V
$\Rightarrow^*$ <<mother_result,sem_mother>$\gg$ <syn_daughter,sem_daughter)>>,sem_result> [$\gg$ E-argument]
if (<<syn_mother,sem_mother>$\gg$ <syn_daughter,sem_daughter)>, sem>, granddaughter)
$\cap >$ <syn_grand,sem_grand >
and T, <syn_grand,sem> $\Rightarrow$ <syn_result,sem_result>

<<(syn_mother,sem_mother>$\gg$ <syn_daughter),sem_daughter)>, sem>, V$\Rightarrow^*$ Z [$\gg$ E-mother]
if (<<syn_mother,sem_mother>$\gg$ <syn_daughter,sem_daughter)>, sem>, mother) $\cap >$ Z

<<syn_mother,sem_mother> $\gg$ <syn_daughter,sem_daughter>, sem>, V$\Rightarrow^*$ Z [$\gg$ E-daughter]
if (<<syn_mother,sem_mother>$\gg$ <syn_daughter,sem_daughter)>, sem>, daughter) $\cap >$ aux
and (<<aux>,sem>, daughter) $\cap >$ spec_daughter
and spec_daughter, V $\Rightarrow^*$ Z

[11] No introduction rules are presented since these would allow inheritance connectives to be introduced in a proof syntactically, whereas they can only originate lexically (cf. the $\nearrow$-operator in Hepple (1990)): a sequent of the form $A\ B \Rightarrow A \gg B$ would come down to the question whether two unrelated signs could maintain an inheritance relation which is not stipulated in the lexicon.

[12] Inclusion of a notion of *dependency* constituency (Barry and Pickering 1990) excludes strings such as *John loves the* from being a constituent in contrast to the original **M**-calculus.

(8)

john kicks mary
<np,john> <<<np\s,*detrans*(sem)>≫<synt(IV)/np,sem)>>, λxλy.kick(x)(y)> <np,mary>
⇒ <s,kick(mary)(john)> [M-Cut]
if <np,john> <np\ s,λxλy.kick(x)(y)> ⇒* <s,λx.kick(x)(john)> [≫ E-argument] (1)
and <<s,*detrans*(sem)>≫< synt(IV)/np,sem)>,λx.kick(x)(john)> <np,mary>
⇒* <s,kick(mary)(john)> [≫ E-daughter] (2)
if <synt(IV)/np,sem> <np,mary)> ⇒* <s,kick(mary)(john) > [M3/] (3)
if <np,mary> ⇒ <np,mary> [Axiom]

The parser starts with the combination *John* and *kicks* (1). *John* serves as the argument of the intransitive form of *kicks*, resulting in a semantic representation that entails that *John* is the subject argument. Next, the combination of the resulting category with *Mary* is attempted. The intransitive frame does not fit here since there is one more *np* in the input, but the transitive frame does (2). *John kicks* and *Mary* are combined (3). In case the intransitive would apply, detransitivisation would be applied. Since there is no more material in the input, the parser stops.
Windowing commits the parser to the information that is present in the input: constituents that maintain head-argument relations are reduced, so the process is incremental. As a result of the reduction a semantic representation is constructed, so the process is immediate. However, the parser does not commit itself to information it has not yet access to. Therefore, erroneous parses are prevented.

5 Lexical preferences and the hierarchical lexicon

Besides *windowing* an equally important source of information that may be exploited to render the interpretation process more efficient in case of ambiguity are *lexical preferences*. To indicate the importance of lexical preferences, the present section opens with a short discussion of preferences as they have been proposed in the literature. Next, lexical preferences are modelled. They follow from the structure of the lexicon, which was independently motivated in order to capture linguistic generalisations. Inference rules model the proceedings of the parser in this respect. Heuristic information is thus integrated in a principled and formal way into the interpretation process. The behaviour of idiomatic expressions will be discussed as an example.

5.1 Preference strategies

Several preference strategies have been proposed for guiding parsers. Amongst these are *structural, syntactic* preferences like Right Association (Kimball 1973), which entails that a modifier should preferably be attached to the rightmost verb(phrase) or noun(phrase) it can modify, and Minimal Attachment (Frazier and Fodor 1979), which states that the analysis which assumes the minimal number of nodes in the syntactic tree should be preferred.[13]
Semantic preferences are illustrated in (9) and (10). The modifiers in both cases are preferably attached contrary to expectations on the basis of syntactic preferences (see Schubert 1984, 1986; Wilks et al. 1985.).

(9) John met the girl that he married at the dance.

(10) John saw the bird with the red beak.

Evidence for the existence of preferences based upon *contextual* information has been provided by Marslen-Wilson and Tyler (1980), who have shown in a number of psycholinguistic experiments that contextual information influences word recognition (see also Crain and Steedman (1982) and Taraban and McClelland (1988)).
Lexical preferencing (Ford et al. 1982) refers to the preference functor categories have for certain arguments. For instance, the verb *to go* can either occur as an intransitive verb that can be modified

[13] See also Shieber (1983) and Hobbs and Bear (1990).

by a *pp* with the prosodic form $to + X$, or it can take this *pp* as an argument. The second frame is the preferred frame. The prepositional phrase should preferably be considered as an argument to the verb and not as a *vp*-modifier.

Although the existence of all of these preferences should thus be acknowledged, there are two arguments in favour of lexical preferences. Firstly, from empirical, corpus-based studies it may be concluded that lexical preferences are successful heuristics for resolving ambiguity (Whittemore et al. (1990); Hobbs and Bear (1990)). Secondly, although ambiguities may be resolved at any level of processing, lexical processing takes place on a lower level since higher levels depend upon lexical information. Resolution of ambiguity on a low level ensures that higher levels of processing are not bothered with ambiguities occurring on lower levels. Therefore, if it is equally possible to model the behaviour of the parser as a lexically guided or as for instance a contextually guided process, the former should be preferred. For instance, in the case of an idiomatic expression, it is more efficient to decide that the idiom should be interpreted on the basis of the mere fact that it is an idiom, than on the basis of consultation of, for instance, some model of the context. Since lexical preferences are successful heuristics that operate on a low level, there is sufficient reason to model them in a principled and formal way.

5.2 Formalisation of lexical preferences

The formalisation of lexical preferences proposed here is another application of the principle of priority to the instance (Hudson 1980): the parser prefers information lower on in the hierarchical structure of the lexicon over information on higher levels in the hierarchy. If two subcategorisation frames of for instance *go* maintain an inheritance relation $< np\backslash s \gg (np\backslash s)/ < pp, to + \_, \_ >>$, and both apply, the more specific frame is preferred. The difference between windowing and lexical preferencing is that windowing applies to the choice during incremental processing among a number of frames of which only one applies eventually, whereas lexical preferencing applies to a choice between frames all of which apply. Lexical preferences do not follow as some statistically motivated preference, but as a linguistically motivated one: lexical preferences follow from the application of the principle of priority to the instance to the use of the structured lexicon.
As was the case with windowing, lexical preferencing can be modelled by means of the inference rules that operate upon inheritance connectives. The implementation of this preference is quite simple. The rules for elimination of the $\gg$-operator are ordered in such a way that the inference engine firstly uses the category as a functor, and next as the argument of a modifier (see definition 8; $A << B$ denotes that A should be applied before B).

Definition 8 (Order of application for $\gg$)

[$\gg$ E-argument] << [$\gg$ E-mother] << [$\gg$ E-daughter]

Note that the boolean operator $\wedge$ does not enable the implementation of this kind of preference. It is, of course, possible to order the categories $((np\backslash s)/ < pp, to > \wedge(np\backslash s))$, and to order the rules that eliminate boolean connectives (first category first). However, the order of these categories must be stipulated, whereas in the case of the hierarchical lexicon structure presented here, the relation between the categories is linguistically motivated. Frequency of occurrence, that is, giving forms with higher frequency prevalence over those with lower frequency, is not an alternative either: more specific forms do not necessarily appear more frequently than the forms they inherit from.

Examples Schubert (1984, 1986) presents a number of sentences which he claims show a preference for attachment that he claims cannot be explained on the basis of structural, syntactic preferences. The preference to attach, for example, $< pp, from + \_ >$ to *disappearance* can, however, be modelled as a lexical preference if *disappearance* (as well as *disappear*) (optionally) subcategorizes for this prepositional phrase. The form with the *pp* then prevails over the form without the *pp*. The same argument applies to (12-15; daughter categories are fully specified).

(11) John was alarmed by the disappearance of the administrator from head office.
disappearance: $n \gg (n/ < pp, from + \_ > / < pp, of + \_ >$

(12) John discussed the girl that he met with his mother.
discuss: $((np\backslash s)/np \gg ((np\backslash s)/ < pp, with + \_ >)/np$

(13) John abandoned the attempt to please Mary.
attempt: $n \gg (n/ < np, to + \_ > \backslash s_{to\_inf} >)$

(14) Sue had difficulties with her teachers.
difficulties: $n \gg (n/ < pp, with + \_ >)$

(15) a. John met the girl that he married at a dance.
b. John married the girl that he met at a dance.
marry: $((np\backslash s)/np)$
met $((np\backslash s)/np) \gg (((np\backslash s)/pp)/np)$

5.3 Idioms and parsing preferences

5.3.1 Conventionality and idiom processing

Idiomatic expressions *can* in most cases be interpreted non-idiomatically as well.[14] It has, however, frequently been observed that an idiomatic phrase should very rarely be interpreted non-idiomatically (Koller 1977:13; Chafe 1968:123; Gross 1984:278; Swinney 1981:208). Also, psycholinguistic research indicates that in case of ambiguity there is clear preference for the idiomatic reading (Gibbs 1980; Schraw et al. 1988; Schweigert 1986; Schweigert and Moates 1988). The phenomenon that phrases should be interpreted according to their idiomatic, noncompositional, lexical, conventional meaning, will be referred to as the 'conventionality' principle (Gibbs 1980). The application of this principle is not limited to idioms. For instance compounds are not interpreted compositionally, but according to the lexical, conventional meaning (Swinney 1981). Words are formed by regular rules, but their meaning will undergo 'semantic drift', obscuring the compositional nature of the complex word.
If this principle could be modelled in an appropriate way, this would be of considerable help in dealing with idioms. As soon as the idiom has been identified, the ambiguity can be resolved and 'higher' knowledge sources do not have to be used to solve the ambiguity. In Stock's (1989) approach to ambiguity resolution the idiomatic and the non-idiomatic analysis are processed in parallel. An external scheduling function gives priority to one of these analyses. Higher knowledge sources are thus necessary to decide upon the interpretation. In PHRAN (Wilensky and Arens 1980), specificity plays a role, but only in suggesting patterns that match the input: evaluation takes place on the basis of *length*, and *order* of the patterns. Zernik and Dyer (1987) present lexical representations for idioms, but do not discuss ambiguity. Van der Linden and Kraaij (1990) discuss two alternative formalisations for conventionality. One extends the notion *continuation class* from two-level morphology. The other is a simple localist connectionist model. Here, another model will be presented which is based upon the specificity of information in the hierarchical structure of the lexicon.

[14] Exceptions are idioms that contain words that occur in idioms only (*spic and span*, *queer the pitch*), and idioms the syntactic form of which is limited to the idiom (*trip the light fantastic*).

5.3.2 Conventionality and the hierarchical lexicon

The ordering of rules for the $\gg$-operator can also be applied to the $\succ$-operator, which relates idioms to verbs. Upon encountering a situation where the $\succ$-operator should be removed, the specific information, the daughter, takes precedence over the more general information, the mother (9).

Definition 9 (Order of application for $\succ$)

[$\succ$ E-daughter] << [$\succ$ E-mother]

The two reduction rules then are presented as in (10). [15]

Definition 10 ($\succ$ -E)

<<(syn_mother,sem_mother>$\succ$ <syn_daughter),sem_daughter)>, sem>, V$\Rightarrow^*$ Z [$\succ$ E-mother]
if (<<syn_mother,sem_mother>$\succ$ <syn_daughter,sem_daughter)>, sem>, mother) $\cap$ > type
and type,V $\Rightarrow^*$ Z

<<syn_mother,sem_mother> $\succ$ <syn_daughter,sem_daughter>, sem>, V$\Rightarrow^*$ Z [$\succ$ E-daughter]
if (<<syn_mother,sem_mother>$\succ$ <syn_daughter,sem_daughter)>, sem>, daughter) $\cap$ > aux
and (<<aux>,sem>, daughter) $\cap$ > type
and type, V $\Rightarrow^*$ Z

As was stated in section 5.2, the boolean operator $\wedge$ does not enable the implementation of this kind of preference. Neither is it possible to model this kind of preference with the use of frequency of occurrence of these forms. On the contrary, since verbs occur within all idioms they are part of, and also occur independently of idioms, their frequency will always be higher than that of the idiomatic expression. Therefore, verbs would always be preferred over idioms, exactly the reverse of what is desired. Also in the case the occurrences of the verb within the idiom are not counted as occurrences of the verb proper, it will be unlikely that on the basis of the frequency criterion the idiom will in all cases be preferred over the verb.
An example of the proceedings of the parser will be presented now in order to illustrate the way windowing, incremental processing and lexical preferences interact in the case of an idiomatic expression. The sign that represents the idiom is abbreviated as k_t_b (compare example 2).
(1) After the lexicalisation of *John* and *kicked*, it becomes possible to form a flexible constituent on the basis of these two words. The result of this step is that, semantically, *John* is considered the subject of any of the verbs in the *kick* hierarchy.
(2) Upon encountering *bucket*, firstly *the* and *bucket* are reduced to an *np* with a prosodic representation *the* + *bucket*. Now it becomes possible to descend in the *kick* hierarchy.
(3) Firstly the choice between the transitive and the intransitive form is made.
(4) Next the choice between the non-idiomatic and the idiomatic form is made.
The derivation results in the assignment of the meaning die(john) to this sentence.

[15] In case a verb occurs in more than one idiomatic expression, for instance *kick the bucket* and *kick one's heels*, only the idiomatic expression that is possible on the basis of the input is used.

(16)

john kicks the bucket.
<np,john> <<<(np\ s),*detrans*(sem)>≫< synt(IV)/np,sem≻ <k_t_b>>,λxλy.kick(x)(y)> <np/n,the> <n,bucket>
⇒ <s,die(john)> [Cut-M] (1)
if <np,john> <np\ s,λxλy.kick(x)(y)> ⇒* <s,λx.kick(x)(john) > [≫-argument]
and <<<s,*detrans*(sem)>≫< synt(IV)/np,sem≻ <k_t_b>>,λx.kick(x)(john)> <np/n,the> <n,bucket>
⇒ <s,die(john)> [M-Cut] (2)
if <np/n,the> <n,bucket> ⇒* <np,the(bucket)> [M3/]
and <<<s,*detrans*(sem)>≫< synt(IV)/np,sem≻ <k t_b>>,λx.kick(x)(john)> <np,the(bucket)>
⇒ <s,die(john)> [M-Cut]
if <<<s,*detrans*(sem)>≫< synt(IV)/np,sem≻ <k_t_b>>,λx.kick(x)(john)> <np,the(bucket)>
⇒* <s,die(john)> [≫ E-daughter](3)
if <synt(IV)/np,sem≻ <k_t_b>> <np,the(bucket)> ⇒* <s,die(john> [≻ E-daughter](4)
if <k_t_b> <np,the(bucket)> ⇒* <s,die(john)> [M3/]
if <np,the(bucket)> ⇒ <np,the(bucket)> [Axiom]
and <s,die(john)> ⇒ <s,die(john)> [Axiom]

5.4 Determinism

Windowing and Lexical Preferencing are nondeterministic processes. Although the parser commits itself only to information it is certain of, and leaves other choices implicit in the structure of the lexicon until it is able to choose (windowing), it can mistake a *vp*-modifier for an argument. Lexical Preferencing is also a nondeterministic process in that backtracking is necessary when interpretations do not fit in the context. Although it is a linguistically motivated strategy, it does not guarantee that the correct choice is made in all cases. In (17) the idiomatic reading is preferred, but later on in the input it turns out that this is not the correct interpretation. Yet, Marcus' *Determinism Hypothesis* states that "(...) all sentences *which people can parse without conscious difficulty* can be parsed strictly deterministically" (Marcus 1980:6). It remains to be seen whether people do not garden-path in (17). Note also that backtracking is modelled very easily - it amounts to making a different choice between two items that maintain an inheritance relation.

(17) John kicked the bucket and Mary the small pail.

6 Implementation

The parser described here has been implemented with the use of a slightly modified version of the categorial calculi interpreter described in Moortgat (1988). This interpreter takes the rules of a calculus as data and applies these recursively to the sequent associated with the input in order to prove that it is a theorem of the calculus. The system is written in Quintus Prolog. No empirical studies of the efficiency of the system have been undertaken so far.

7 Concluding remarks

The hierarchical structure of the lexicon can make a contribution to the speed and the efficiency of the resolution of ambiguity during the process of understanding natural language. With the use of other connectives, or other properties of lexical items like frequency, it is not possible to model this. The hierarchical lexicon should thus not only be considered as vital for the reduction of redundancy in the computational lexicon, or as an aid for developing large lexicons, but also as a source for rendering the parsing process faster and more efficient.
The lexicalism and representational nonautonomy of categorial grammar enable a principled and formal way to model the proceedings of a 'lexicon-sensitive' parser. Categorial rules not only model how categories are combined to form other categories, but also represent parsing in the case of lexical ambiguities. The order in which the inference rules are used implements the preferences of the parser.

Proper inclusion precedence seems to apply in generation too, except that semantic instead of syntactic

hierarchies should be used. During the generation of a sentence containing a collocation, *John commits a murder*, the appropriate verb has to be generated on the basis of the noun. Since *commit* is more specific than, for instance, *do* or *make* in that it subcategorizes for criminal acts and the like, *commit* is selected. Application to generation is possible for Categorial Grammar: the Lambek-calculus can be used bidirectionally, and the theorem proving framework is a natural candidate for a uniform processing architecture (van der Linden and Minnen 1990).
Although representational nonautonomy is not a principle that applies to other frameworks, there seems to be no objection to extend some of these frameworks. For instance, besides the substitution and the adjunction operation of TAG, other, 'lexicon-sensitive' tree-forming operations could be added. Therefore, the approach taken here might carry over to other frameworks.

8 Acknowledgments

Thanks to Walter Daelemans for his continuous plea in favour of the hierarchical lexicon. Without it, I would not have started the research reported on in this article. Thanks are also due to Michael Moortgat for arousing my interest in categorial logic and for his valuable feedback on all aspects of it. Gosse Bouma's introduction of default unification in CG initiated my thinking about the application of inheritance to idioms. Thanks to Harry Bunt, Koenraad De Smedt, Martin Everaert, Hans Kerkman, Glynn Morrill, André Schenk, Carl Vogel, Ton van der Wouden, and three CL referees for comments, suggestions, and discussion. Michael Moortgat generously supplied a copy of the categorial calculi interpreter described in his 1988 thesis. André Schenk and Mark Hepple provided some of the LaTeXmacro's used. Part of the research in this article has been supported by a grant from the Netherlands Organisation for Scientific Research (NWO).

References

Abeillé, A. 1990 Lexical and Syntactic Rules in a Tree Adjoining Grammar. In *Proceedings of ACL 1990*, 292-298.

Abeillé, A. and Schabes, Y. 1989 Parsing Idioms in Lexicalized TAGs. In *Proceedings of EACL 1989*, 1-9.

Ades, A. and Steedman, M. 1982 On the Order of Words. *Linguistics and Philosophy* 4: 517-558.

Adriaens, G. 1986 *Process Linguistics.* Ph.D. thesis. University of Leuven.

Aronoff, M. 1976 *Word Formation in Generative Grammar.* The MIT Press, Cambridge, Massachusetts.

Barry, G. and Morrill, G. (eds.) 1990 *Studies in Categorial Grammar.* University of Edinburgh.

Barry, G. and Pickering, M. 1990 Dependency and Constituency in Categorial Grammar. In: Barry and Morrill (eds.), 23-45.

Benthem, J. van 1986 Categorial Grammar. In Van Benthem, J. 1986 *Essays in Logical Semantics.* Reidel, Dordrecht, Chapter 7.

Bouma, G. 1989 Efficient Processing of Flexible Categorial Grammar. In *Proceedings of EACL 1989*, 19-26.

Bouma, G. 1990 Defaults in Unification Grammar, In *Proceedings of ACL 1990*, 165-172.

Briscoe, E. 1987 *Modelling Human Speech Comprehension.* Ellis Horwood, Chichester.

Chafe, W. 1968 Idiomaticity as an Anomaly in the Chomskyan Paradigm. *Foundations of Language* 4, 109-127.

Crain, S. and Steedman, M. 1982 On not being led up the Garden Path. In Dowty, D.; Karttunen, L.; and Zwicky, A. (eds.) *Natural Language Parsing*, Cambridge University Press, Cambridge, 320-358.

Daelemans, W. 1987 *Studies in Language Technology: an Object-oriented Model of Morphophonological Aspects of Dutch.*, Ph.D. thesis, University of Leuven.

De Smedt, K. 1990 *Incremental Sentence Generation.* Ph.D. thesis, University of Nijmegen.

Dowty, R. 1979 *Word Meaning and Montague Grammar.* Reidel, Dordrecht.

Dowty, R. 1988 Type Raising, Functional Composition and Non-constituent Conjunction. In Oehrle et al. (eds.), 153-197.

Erbach, G. 1991 Lexical Representation of Idioms. IWBS report 169, IBM TR-80.91-023, IBM, Germany.

Flickinger, D. 1987 *Lexical Rules in the Hierarchical Lexicon.* Ph.D. thesis, Stanford University.

Ford, M.; Bresnan, J. and Kaplan, R. 1982 A Competence-based Theory of Syntactic Closure. In Bresnan, J. (ed.) *The mental Representation of Grammatical Relations.* The MIT Press, Cambridge, Massachusetts.

Frazier, L. and Fodor, J. 1978 The Sausage Machine: a New Two-stage Parsing Model. *Cognition* 6: 291-325.

Gibbs, R. 1980 Spilling the Beans on Understanding and Memory for Idioms in Conversation. *Memory and Cognition* 8: 149-156.

Gross, M. 1984 Lexicon-grammar and the Syntactic Analysis of French. In *Proceedings of COLING 1984*, 275-282.

Haddock, N. 1987 Incremental Interpretation and Combinatory Categorial Grammar. In Haddock et al. (eds.), 71-84.

Haddock, N.; Klein, E. and Morrill, G. (eds.) 1987 *Working Papers in Cognitive Science, Volume 1. Categorial Grammar, Unification Grammar and Parsing.* Centre for Cognitive Science, University of Edinburgh.

Hendriks, H. 1987 Type Change in Semantics: the Scope of Quantification and Coordination. In: Klein and van Benthem (eds.), 95-119.

Hepple, M. 1990 Word Order and Obliqueness in Categorial Grammar. In Barry and Morrill (eds.), 47-64.

Hirst, G. 1988 "Resolving Lexical Ambiguity Computationally with Spreading Activation and Polaroid Words. In Small et al. (eds.), 73-107.

Hobbs, J. and Bear, J. 1990 Two Principles of Parse Preference. In *Proceedings of COLING 1990*, 162-167.

Houtman, J. 1987 Coordination in Dutch. In Klein and van Benthem (eds.), 121-145.

Hudson, R. 1984 *Word Grammar.* Blackwell: Oxford.

Just, M. and Carpenter, P. 1980 A Theory of Reading, from Eye Fixations to Comprehension. *Psychological Review* 87: 329-354.

Kaplan, R. 1987. Three Seductions of Computational Psycholinguistics. In Whitelock, P.; Wood, M. McGee; Somers, H.; Johhson, R., and Bennett, P. (eds.) *Linguistic Theory and Computer Applications* Acacemic Press, London, 149-188.

Kempen, G. and Vosse, Th. 1989 Incremental Syntactic Tree Formation in Human Sentence Processing: a Cognitive Architecture Based on Activation Decay and Simulation Annealing. *Connection Science* 1: 275-292.

Kimball, J. 1973 Seven Principles of Surface Structure Parsing in Natural Language. *Cognition* 2: 15-47.

Klein, E. and van Benthem, J. (eds.) *Categories, Polymorphism and Unification.* Centre for Cognitive Science, University of Edinburgh, Institute for Language, Logic and Information, University of Amsterdam.

Koller, W. 1977 *Redensarten: Linguistische Aspekte, Vorkommensanalysen, Sprachspiel.* Tübingen, Niemeyer.

Lambek, J. 1958 The Mathematics of Sentence Structure. *Am. Math. Monthly* 65: 154-169.

Leslie, N. 1990 Contrasting Styles of Categorial Derivations. In Barry and Morrill (eds.), 113-126.

Van der Linden, E. 1991 Idioms, Non-literal Language and Knowledge Representation. In *Proceedings of the IJCAI-workshop Computational approaches to non-literal language.*

Van der Linden, E. and Kraaij, W. 1990 Ambiguity Resolution and the Retrieval of Idioms: Two Approaches. In *Proceedings of COLING 1990*, vol 2: 245-251.

Van der Linden, E. and Minnen, G. 1990 Algorithms for Generation in Lambek Theorem Proving. In *Proceedings of ACL 1990*, 220-226.

Marcus, M. 1980 *A Theory of Syntactic Recognition for Natural Language.* MIT Press, Cambridge, Massachusetts.

Marslen-Wilson, W. and Tyler, L. 1980 The Temporal Structure of Spoken Language Understanding. *Cognition* 8: 1-71.

Moortgat, M. 1987 Lambek Theorem Proving. In Klein and van Benthem (eds.), 169-200.

Moortgat, M. 1988 *Categorial Investigations, Logical and Linguistic Aspects of the Lambek Calculus.* Ph.D. thesis, University of Amsterdam.

Moortgat, M. (1990) Categorial Logics: a Computational Perspective. In: Computation in the Netherlands. Edited by A.J. van Goor, 329-347.

Moortgat, M. (1992) The logic of Discontinuous Type Constructors. In Sijtsma, W. and Horck, A. van (eds.) *Discontinuous Constituency.* Mouton de Gruyter, Berlin.

Morrill, G. 1990 Grammar and Logical Types. In Barry and Morrill (eds.), 127-148.

Morrill, G.; Leslie, N.; Hepple, M.; and Barry, G. 1990 Categorial Deductions and Structural Operations. In Barry and Morrill (1990), 1-21.

Oehrle, R.; Bach, E. and Wheeler, D. (eds). 1988 *Categorial Grammar and Natural Language Structure.*, Reidel, Dordrecht.

Pollard and Sag 1987 *Information-based Syntax and Semantics* Vol. 1 CSLI, Stanford.

Ristad, E. 1990 *Computational Structure of Human Language.* Ph.D. thesis, MIT Department of Electrical Engineering and Computer Science.

Roorda, D. 1990 Proofnets for Lambek Calculus. Ms. University of Amsterdam.

Schraw, G.; Trathen, W.; Reynolds, R. and Lapan R. 1988 Preferences for Idioms: Restrictions due to Lexicalization and Familiarity. *Journal of Psycholinguistic Research* 17: 413-424.

Schubert, L. 1984 On Parsing Preferences. In *Proceedings of COLING 1984*, 247-250.

Schubert, L. 1986 Are there Preference Trade-offs in Attachment Decisions? In *Proceedings of AAAI-86*, 601-605.

Schweigert, W. 1986 The Comprehension of Familiar and Less Familiar Idioms. *Journal of Psycholinguistic Research* 15: 33-45.

Schweigert, W. and Moates, D., 1988 Familiar Idiom Comprehension. *Journal of Psycholinguistic Research* 17: 281-296.

Shieber, S. 1983 Sentence Disambiguation by Shift-reduce Parsing Technique. In *Proceedings of IJCAI 1983*, 699-703.

Small, S.; Cottrell, G. and Tanenhaus, M. (eds.) 1988 *Lexical Ambiguity Resolution.* Kaufmann, San Mateo.

Steedman, M. 1987 Combinatory Grammars and Parasitic Gaps. In Haddock et al. (eds.), 30-70.

Stock, O. 1989 Parsing with Flexibility, Dynamic Strategies, and Idioms in Mind. *Computational Linguistics* 15: 1-19.

Swinney, D. 1979 Lexical Access during Sentence Comprehension: (re)consideration of context effects. *Journal of Verbal Learning and Verbal Behaviour* 18: 645-659.

Swinney, D. 1981 Lexical Processing during Sentence Comprehension: Effects of Higher Order Constraints and Implications for Representation. In Meyers, T.; Laver, J.; and Anderson, J. (eds.) *The Cognitive Representation of Speech*, North-Holland.

Taraban, R. and McClelland, J. 1988 Constituent Attachment and Thematic Role Assignment in Sentence Processing: Influences of Content-based Expectations. *Journal of Memory and Language* 27: 597-632.

Touretzky, D. 1986 *The Mathematics of Inheritance Systems.* Morgan Kaufmann Publishers, Los Altos, CA.

Touretzky, D.; Horty, J. and Thomason, R. 1987 A Clash of Intuitions: the Current State of Non-monotonic Multiple Inheritance Systems. *Proceedings of IJCAI 1987*, 476-482.

Veltman, F. 1990 Defaults in Update Semantics I. In H. Kamp (ed.) Conditionals, Defaults and Belief Revision. DYANA Deliverable R2.5.A, 28-63.

Whittemore, G.; Ferrara, K. and Brunner, H. 1990 Empirical Study of Predictive Powers of Simple Attachment Schemes for Post-modifier Prepositional Phrases. In *Proceedings of ACL 1990* 23-30.

Wilensky, R. and Arens, Y. 1980 PHRAN, A Knowledge-based Natural Language Understander. In *Proceedings of the ACL 1980*, 117-121.

Wilks, Y.; Huang, X. and Fass, D. 1985 Syntax, Preference and Right Attachment. In *Proceedings of IJCAI 1985*, 779-784.

Zernik, U. and Dyer, M. 1987 The Self-extending Phrasal Lexicon, *Computational Linguistics* 13: 308-327.

OVERVIEW OF ITK RESEARCH REPORTS

No	Author	Title
1	H.C. Bunt	On-line Interpretation in Speech Understanding and Dialogue Sytems
2	P.A. Flach	Concept Learning from Examples Theoretical Foundations
3	O. De Troyer	RIDL*: A Tool for the Computer-Assisted Engineering of Large Databases in the Presence of Integrity Constraints
4	M. Kammler en E. Thijsse	Something you might want to know about "wanting to know"
5	H.C. Bunt	A Model-theoretic Approach to Multi-Database Knowledge Representation
6	E.J. v.d. Linden	Lambek theorem proving and feature unification
7	H.C. Bunt	DPSG and its use in sentence generation from meaning representations
8	R. Berndsen en H. Daniels	Qualitative Economics in Prolog
9	P.A. Flach	A simple concept learner and its implementation
10	P.A. Flach	Second-order inductive learning
11	E. Thijsse	Partical logic and modal logic: a systematic survey
12	F. Dols	The Representation of Definite Description
13	R.J. Beun	The recognition of Declarative Questions in Information Dialogues
14	H.C. Bunt	Language Understanding by Computer: Developments on the Theoretical Side
15	H.C. Bunt	DIT Dynamic Interpretation in Text and dialogue
16	R. Ahn en H.P. Kolb	Discourse Representation meets Constructive Mathematics

No	Author	Title
17	G. Minnen en E.J. v.d. Linden	Algorithmen for generation in lambek theorem proving
18	H.C. Bunt	DPSG and its use in parsing
19	H.P. Kolb	Levels and Empty? Categories in a Principles and Parameters Approach to Parsing
20	H.C. Bunt	Modular Incremental Modelling Belief and Intention
21	F. Dols	Compositional Dialogue Referents in Prase Structure Grammar
22	F. Dols	Pragmatics of Postdeterminers, Non-restrictive Modifiers and WH-phrases
23	P.A. Flach	Inductive characterisation of database relations
24	E. Thijsse	Definability in partial logic: the propositional part
25	H. Weigand	Modelling Documents
26	O. De Troyer	Object Oriented methods in data engineering
27	O. De Troyer	The O-O Binary Relationship Model
28	E. Thijsse	On total awareness logics
29	E. Aarts	Recognition for Acyclic Context Sensitive Grammars is NP-complete
30	P.A. Flach	The role of explanations in inductive learning
31	W. Daelemans, K. De Smedt en J. de Graaf	Default inheritance in an object-oriented representation of linguistic categories
32	E. Bertino H. Weigand	An Approach to Authorization Modeling in Object-Oriented Database Systems
33	D.M.W. Powers	Multi-Modal Modelling with Multi-Module Mechanisms: Autonomy in a Computational Model of Language

No	Author	Title
34	R. Muskens	Anaphora and the Logic of Change*
35	R. Muskens	Tense and the Logic of Change
36	E.J. v.d. Linden	Incremental Processing and the Hierarchical Lexicon
37	E.J. v.d. Linden	Idioms, non-literal language and knowledge representation 1

ITK: P.O. BOX 90153 5000 LE TILBURG THE NETHERLANDS